# Bài 1: CHUẨN BỊ DỮ LIỆU

## I. Mục tiêu:

Sau khi thực hành xong, sinh viên nắm được:

- Sử dụng thư viện Pandas để đọc file, thư viện scikit-learn.
- Chuẩn hóa dữ liệu
- PCA

## II. Tóm tắt lý thuyết:

Tập dữ liệu nhiều chiều D là một tập hợp gồm n bản ghi $\overline{X_1}, \overline{X_2}, \dots, \overline{X_n}$, sao cho mỗi $\overline{X_\iota}$ là một tập hợp chứa d đặc trưng được ký hiệu bởi $\left(x_i^1, \dots, x_i^d\right)$.

### 1. Chuẩn hóa dữ liệu:

Xét trường hợp thuộc tính thứ j có trung bình (mean) là $\mu_j$ và độ lệch chuẩn (standard deviation) $\sigma_j$ . Khi đó, giá trị thuộc tính thứ j $x_i^j$ của bản ghi thứ i $\overline{X_\iota}$ có thể được chuẩn hóa như sau:

$$z_i^j = \frac{x_i^j - \mu_j}{\sigma_j}$$

Phần lớn các giá trị được chuẩn hóa sẽ nằm trong vùng [-3, 3] dưới giả thuyết phân phối chuẩn.

Xấp xỉ thứ 2 sử dụng min-max scaling để ánh xạ tất cả thuộc tính thành vùng [0,1]. Đặt $min_j$ và $max_j$ biểu diễn các giá trị nhỏ nhất và lớn nhất của thuộc tính j. Khi đó, giá trị thuộc tính j $x_i^j$ của bản ghi thứ i $\overline{X_\iota}$ có thể được scaled như sau:

$$y_i^j = \frac{x_i^j - min_j}{max_j - min_j}$$

Xấp xỉ này không hiệu quả khi các giá trị lớn nhất và giá trị nhỏ nhất là các giá trị ngoại lệ cực biên bởi vì một vài lỗi trong thu thập dữ liệu. Ví dụ, xét thuộc tính tuổi mà lỗi trong thu thập dữ liệu gây ra số 0 thêm vào để được cộng thêm vào một tuổi, dẫn đến giá trị tuổi sẽ nằm trong vùng [0,0.1], vì kết quả thuộc tính này có thể được chỉnh giảm.

2. **Equi-width ranges:** Trong trường hợp này, mỗi vùng [a,b] được chọn trong cách mà b-a là giống nhau cho mỗi vùng. Xấp xỉ này có hạn chế là sẽ không làm việc cho các

tập dữ liệu được phân phối không đều nhau qua các vùng khác nhau. Để xác định các giá trị chính xác của các vùng, các giá trị nhỏ nhất và lớn nhất của mỗi thuộc tính (attribute) được xác định. Khi đó, vùng này [min, max] được chia thành $\phi$ vùng có chiều dài bằng nhau.

3. **Equi-depth ranges:** Trong trường hợp này, các vùng được được chọn sao cho mỗi vùng có số các bản ghi bằng nhau. Một thuộc tính có thể được chia thành các vùng equi-depth bằng việc sắp xếp nó đầu tiên và sau đó việc lựa chọn các điểm phân chia trong giá trị thuộc tính được sắp xếp, sao cho mỗi vùng chứa số các bản ghi bằng nhau.
4. **PCA:**

   PCA thường được dùng sau khi áp dụng mean centering cho dữ liệu, tức là mỗi điểm trong dữ liệu trừ đi trung bình dữ liệu. Khi đó, tâm của tập dữ liệu sẽ ở góc tọa độ

   Ngoài ra, nếu trung bình dữ liệu được lưu riêng thì cũng có thể áp dụng PCA mà không cần mean centering.

   Cho C là ma trận hiệp phương sai đối xứng $d \times d$ của ma trận dữ liệu D $n \times d$. Khi đó, phần tử $c_{ij}$ của C ký hiệu hiệp phương sai giữa cột i và cột j (số chiều) của ma trận dữ liệu D. Cho $\mu_i$ biểu diễn trung bình theo chiều thứ i. Rõ ràng, nếu $x_k^m$ là bản ghi thứ k của chiều thứ m thì giá trị của phần tử hiệp phương sai $c_{ij}$ được tính như sau:

$$c_{ij} = \frac{\sum_{k=1}^{n} x_k^i x_k^j}{n} - \mu_i \mu_j \quad \forall i, j \in \{1, \dots, d\}$$

   Cho $\bar{\mu} = (\mu_1 \dots \mu_2)$ là vector dòng d chiều biểu diễn các giá trị trung bình theo các chiều khác nhau. Khi đó, các tính toán $d \times d$ ở trên của phương trình cho các giá trị khác nhau của i và j có thể được biểu diễn trong dạng ma trận $d \times d$ như sau:

$$C = \frac{D^T D}{n} - \bar{\mu}^T \bar{\mu}$$

   Chú ý rằng d phần tử đường chéo của ma trận C tương ứng với d phương sai. Ma trận hiệp phương sai C nửa xác định dương, bởi vì nó có thể được chứng minh rằng với vector cột d chiều $\bar{v}$, giá trị của $\bar{v}^T C \bar{v}$ bằng với phương sai của phép chiếu 1 chiều $D\bar{v}$ của tập dữ liệu D trong $\bar{v}$.

$$\bar{v}^T C \bar{v} = \frac{(D\bar{v})^T D\bar{v}}{n} - (\bar{\mu}\bar{v})^2 = phương\ sai\ của\ các\ điểm\ 1\ chiều\ trong\ D\bar{v} \geq 0$$

Thật vậy, mục tiêu của PCA là để xác định liên tục các vector trực giao $\bar{v}$ cực đại $\bar{v}^T C \bar{v}$. Bởi vì ma trận hiệp phương sai là đối xứng và nửa xác định dương nên nó có thể được chéo hóa như sau:

$$C = P\Lambda P^T$$

Các cột của ma trận P chứa các vector trực giao của C, Λ là ma trận đường chéo chứa các giá trị riêng không âm. Phần tử $\Lambda_{ii}$ là trị riêng tương ứng với vector riêng thứ i (hoặc cột ) của ma trận P. Các vector riêng này biểu diễn các lời giải trực giao liên tục nhau thành mô hình tối ưu hóa cực đại phương sai $\bar{v}^T C \bar{v}$ theo phương thống nhất $\bar{v}$.

Một tính chất thú vị của sự chéo hóa này là cả vector riêng và trị riêng đều có một thể hiện hình học liên quan tới phân phối dữ liệu cơ bản. Đặc biệt, nếu hệ trục của biểu diễn dữ liệu được xoay thành tập trực giao của các vector riêng trong các cột của P thì nó có thể được chứng minh rằng $\binom{d}{2}$ hiệp phương sai của các giá trị đặc trưng được biến đổi mới là 0. Mặc khác, các phương lưu trữ phương sai lớn nhất cũng là các phương tương quan loại bỏ. Hơn nữa, các trị riêng biểu diễn các phương sai của dữ liệu theo các vector riêng tương ứng. Thật vậy, ma trận đường chéo Λ là ma trận phương sai mới sau khi xoay trục. Do đó, các vector riêng với các trị riêng lớn bảo toàn phương sai lớn hơn và cũng được nhắc đến như các thành phần chính. Bởi vì sự tự nhiên của công thức tối ưu hóa thường sử dụng để suy ra sự biến đổi này, một hệ thống trục mới chỉ chứa các vector riêng với các trị riêng lớn nhất được tối ưu thành phương sai lớn nhất liên tục trong một số chiều được cố định.

## III. Nội dung thực hành:

1. Download Arrythmia từ UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.php)


a. Đọc dữ liệu từ file:

- Thêm vào các thư viện và module cần thiết:

```
import pandas as pd
import numpy as np
```

- Cài đặt thư viện scikit-learn:

```
C:\Users\Huynh>pip install -U scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-1.0.2-cp37-cp37m-win_amd64.whl (7.1 MB)
     ---------------------------------------- 7.1/7.1 MB 4.9 MB/s eta 0:00:00
Requirement already satisfied: scipy>=1.1.0 in c:\users\huynh\appdata\local\programs\python\python37\lib\site-packages (from scikit-learn) (1.3.1)
Requirement already satisfied: numpy>=1.14.6 in c:\users\huynh\appdata\local\programs\python\python37\lib\site-packages (from scikit-learn) (1.17.2)
Collecting joblib>=0.11
  Downloading joblib-1.2.0-py3-none-any.whl (297 kB)
     ---------------------------------------- 298.0/298.0 kB 2.3 MB/s eta 0:00:00
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-3.1.0-py3-none-any.whl (14 kB)
Installing collected packages: threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.2.0 scikit-learn-1.0.2 threadpoolctl-3.1.0
```

- Đọc dữ liệu từ file:

```
>>> url='https://archive.ics.uci.edu/ml/machine-learning-databases/arrhythmia/arrhythmia.data'
>>> df = pd.read_csv(url)
>>> df.head()
   75  0  190  80   91  193  371  ...  0.0.39  0.0.40  0.9.3 2.9.1  23.3  49.4   8
0  56  1  165  64   81  174  401  ...     0.0     0.0    0.2   2.1  20.4  38.8   6
1  54  0  172  95  138  163  386  ...     0.0     0.0    0.3   3.4  12.3  49.0  10
2  55  0  175  94  100  202  380  ...     0.0     0.0    0.4   2.6  34.6  61.6   1
3  75  0  190  80   88  181  360  ...     0.0     0.0   -0.1   3.9  25.4  62.8   7
4  13  0  169  51  100  167  321  ...     0.0     0.0    0.9   2.2  13.5  31.1  14

[5 rows x 280 columns]
>>> df.tail()
     75  0  190  80   91  193  ...  0.0.40  0.9.3  2.9.1  23.3  49.4   8
446  53  1  160  70   80  199  ...     0.0    0.7    0.6  -4.4  -0.5   1
447  37  0  190  85  100  137  ...     0.0    0.4    2.4  38.0  62.4  10
448  36  0  166  68  108  176  ...     0.0    1.5    1.0 -44.2 -33.2   2
449  32  1  155  55   93  106  ...     0.0    0.5    2.4  25.0  46.6   1
450  78  1  160  70   79  127  ...     0.0    0.5    1.6  21.3  32.8   1

[5 rows x 280 columns]
```

- Trung bình (mean), độ lệch chuẩn (standard deviation), min, max

```
>>> df.describe()
                0           1           2    ...         277         278         279
count  452.000000  452.000000  452.000000  ...  452.000000  452.000000  452.000000
mean    46.471239    0.550885  166.188053  ...   19.326106   29.473230    3.880531
std     16.466631    0.497955   37.170340  ...   13.503922   18.493927    4.407097
min      0.000000    0.000000  105.000000  ...  -44.200000  -38.600000    1.000000
25%     36.000000    0.000000  160.000000  ...   11.450000   17.550000    1.000000
50%     47.000000    1.000000  164.000000  ...   18.100000   27.900000    1.000000
75%     58.000000    1.000000  170.000000  ...   25.825000   41.125000    6.000000
max     83.000000    1.000000  780.000000  ...   88.800000  115.900000   16.000000

[8 rows x 275 columns]
```

- Thay thế "?" bằng giá trị np.NaN

```
>>> df = df.replace('?',np.NaN)
```

b. Chuẩn hóa tất cả các bản ghi có trung bình 0, độ lệch chuẩn 1:

Module preprocessing cung cấp lớp StandardScaler hữu dụng

```
>>> array=df.values
```

```
>>> data_scaler=StandardScaler()
>>> array
array([[75, 0, 190, ..., 23.3, 49.4, 8],
       [56, 1, 165, ..., 20.4, 38.8, 6],
       [54, 0, 172, ..., 12.3, 49.0, 10],
       ...,
       [36, 0, 166, ..., -44.2, -33.2, 2],
       [32, 1, 155, ..., 25.0, 46.6, 1],
       [78, 1, 160, ..., 21.3, 32.8, 1]], dtype=object)
>>> scale = data_scaler.fit_transform(array)
>>> scale
array([[ 1.73443926, -1.1075202 ,  0.64132669, ...,  0.29460309,
         1.07867028,  0.93577103],
       [ 0.57931213,  0.90291807, -0.03199781, ...,  0.0796127 ,
         0.50487408,  0.4814547 ],
       [ 0.4577198 , -1.1075202 ,  0.15653305, ..., -0.52087767,
         1.0570176 ,  1.39008736],
       ...,
       [-0.63661117, -1.1075202 , -0.00506483, ..., -4.7094834 ,
        -3.3926096 , -0.42717797],
       [-0.87979583,  0.90291807, -0.30132761, ...,  0.42063193,
         0.92710147, -0.65433613],
       [ 1.91682776,  0.90291807, -0.16666271, ...,  0.14633386,
         0.18008377, -0.65433613]])
```

c. Rời rạc hóa mỗi thuộc tính số hóa thành

- 10 equi-width ranges
- 10 equi-depth ranges

2. Download the Musk data set from the UCI Machine Learning Repository [213]. Apply PCA to the data set, and report the eigenvectors and eigenvalues.